PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-101045
(43)Date of publication of application : 13.04.2001

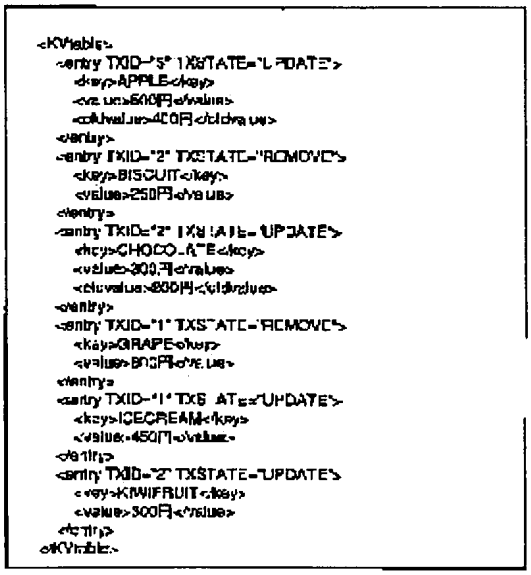
(51)Int.Cl. G06F 12/00

(21)Application number : 11-276264
(22)Date of filing : 29.09.1999

(71)Applicant : TOSHIBA CORP
(72)Inventor : KANAI TATSUNORI
TORII OSAMU
KITSU TOSHIKI
MAEDA SEIJI
YAO HIROSHI
YANO HIROKUNI

(54) METHOD AND SYSTEM FOR PROCESSING TRANSACTION

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a transaction processing system for easily and efficiently realizing file update processing and recovery processing on the occurrence of any failure by plural transactions operating in parallel.
SOLUTION: As for a file in which UPDATE or REMOVE or the like is updated by plural transactions (TXID=1-3) operating in parallel, when one transaction (TXID=1) which updates this file commits, information (TXID=2 and TXSTATE=UPDATE, <oldvalue> 200 yen </oldvalue> or the like in <entry>) necessary for the update of the file by the transaction but also the update of the file by the other transactions (TXID=2, 3) and the cancellation of this is written in the file in a batch. Then, recovery processing after the occurrence of any failure is operated at the point of using the file the next.



LEGAL STATUS

[Date of request for examination] 01.12.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－101045
（P2001－101045A）
(43)公開日　平成13年４月13日(2001.4.13)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　12/00	５１８	Ｇ０６Ｆ　12/00	５１８Ａ　５Ｂ０８２

審査請求　未請求　請求項の数８　ＯＬ　（全 26 頁）

(21)出願番号　特願平11－276264

(22)出願日　平成11年９月29日(1999.9.29)

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　金井　達徳
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(72)発明者　鳥井　修
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(74)代理人　100058479
弁理士　鈴江　武彦　（外６名）

最終頁に続く

(54)【発明の名称】　トランザクション処理方法及びトランザクション処理システム

(57)【要約】
【課題】　並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することのできるトランザクション処理システムを提供すること。
【解決手段】　並行に動作する複数のトランザクション（ＴＸＩＤ＝１～３）によってＵＰＤＡＴＥやＲＥＭＯＶＥなどの更新をされたファイルに対して、それを更新した１つのトランザクション（ＴＸＩＤ＝１）のコミット時に、そのトランザクションによる更新だけでなく、他のトランザクション（ＴＸＩＤ＝２，３）による更新とそれを取り消すのに必要な情報（＜ｅｎｔｒｙ＞におけるＴＸＩＤ＝２とＴＸＳＴＡＴＥ＝ＵＰＤＡＴＥ、＜ｏｌｄｖａｌｕｅ＞２００円＜／ｏｌｄｖａｌｕｅ＞等）を、同じファイルにまとめて書き込む。障害発生後のリカバリ処理は次にそのファイルを使う時点で行えばよい。

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
```

1

【特許請求の範囲】

【請求項１】バッファ領域上に読み出した同一のファイルを更新対象として複数のトランザクションが処理を行い、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、該一つのコミットするトランザクションによる確定した更新内容と、他のコミットしていないトランザクションによる未確定の更新内容と、該未確定の更新を取り消すための情報とをファイル内容として含む前記ファイルを、安定記憶装置に書き込むことを特徴とするトランザクション処理方法。

【請求項２】前記他のコミットしていないトランザクションをコミットする時には、該他のこれからコミットするトランザクションによる新たに確定した更新内容と、以前にコミットしたトランザクションによる既に確定している更新内容のうち該他のこれからコミットするトランザクションによる新たに確定した更新内容とは競合しないものとをファイル内容として含むファイルを、前記安定記憶装置に格納された前記ファイルに上書きすることを特徴とする請求項１に記載のトランザクション処理方法。

【請求項３】新しくファイルを前記安定記憶装置から読み出した時に、このファイルに、コミットしていないトランザクションによる未確定の更新内容および該未確定の更新を取り消すための情報が含まれているか否かを調べ、

含まれている場合には、読み出した前記ファイルを前記情報を用いて前記コミットしていないトランザクションによる更新がなかった状態に戻すことを特徴とする請求項１または２に記載のトランザクション処理方法。

【請求項４】前記一つのコミットするトランザクションが前記ファイルとは別のファイルをも更新対象として前記バッファ領域上で処理を行っていた場合には、前記ファイルの前記安定記憶装置への書き込みと同期させて、該一つのコミットするトランザクションによる前記別のファイルに対する確定した更新内容をファイル内容として含む前記別のファイルを前記安定記憶装置に書き込むことを特徴とする請求項１ないし３のいずれか１項に記載のトランザクション処理方法。

【請求項５】前記ファイルは、一纏まりの情報を分割して記録した複数個のファイルの集合の中の一つのファイルであることを特徴とする請求項１ないし４のいずれか１項に記載のトランザクション処理方法。

【請求項６】一纏まりの情報を分割して記録した複数個のファイルの集合に対して二つ以上のトランザクションが前記バッファ領域上で更新を行い、

この二つ以上のトランザクションによる更新内容に基づいて、前記バッファ領域上で、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファイルそれぞれの大きさが予め定められた基準に従うように調整し、

前記一つのコミットするトランザクションが更新したファイル、および該トランザクションによる更新に伴う前記調整の結果更新あるいは新規に作成されたファイルのうち、少なくとも一つのファイルに対して他のトランザクションが更新を行っている場合には、該少なくとも一つのファイルを更新対象として複数のトランザクションが処理を行ったものとして、前記ファイルの前記安定記憶装置への書き込みを行うことを特徴とする請求項１ないし４のいずれか１項に記載のトランザクション処理方法。

【請求項７】トランザクションの処理対象となるファイルを安定記憶装置からバッファ領域上に読み出す手段と、

前記ファイルに対して複数のトランザクションのそれぞれが行った更新の内容と該更新を取り消すための情報とを前記バッファ領域上で該ファイル内に書き込む手段と、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイル内に書き込まれた他のコミットしていないトランザクションによる更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容とそのままとした情報とをファイル内容として含む前記ファイルを、前記安定記憶装置に書き出す手段とを備えたことを特徴とするトランザクション処理システム。

【請求項８】トランザクションの処理対象となる一纏まりの情報を分割して記録した複数個のファイルの集合を安定記憶装置からバッファ領域上に読み出す手段と、

前記ファイルの集合に対して複数のトランザクションのそれぞれが行った更新の内容に基づいて、前記バッファ領域上で、該更新の内容と該更新を取り消すための情報とをファイル内に書き込むとともに、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファイルそれぞれの大きさが予め定められた基準に従うように調整する手段と、

前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイルおよび前記調整の結果更新あるいは新規に作成されたファイルのうちの少なくとも一つのファイルに対して更新を行った他のコミットしていないトランザクションによる、該少なくとも一つのファイル内に書き込まれた更新の内容および該更新を

取り消すための情報についてはそのままとし、確定させた内容をファイル内容として含む前記ファイルとそのままとした情報をファイル内容として含む前記少なくとも一つのファイルとを、前記安定記憶装置に書き出す手段とを備えたことを特徴とするトランザクション処理システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のトランザクションが並行して同じファイルにアクセスするトランザクション処理方法及びトランザクション処理システムに関する。

【０００２】

【従来の技術】一般に、トランザクション処理システムにおいては、トランザクションと呼ぶ処理の流れを単位として処理の実行を管理する。例えば、端末装置やネットワークなどから受け取った処理要求に対して、必要な処理を実行し、その結果を返すような、何らかの手段で起動された処理を実行する流れをトランザクションと呼ぶ。個々のトランザクションは、実行過程で、１または複数のファイル（あるいはデータベース）に記録管理するデータの更新を行う。トランザクション処理システムは、このような１または複数のデータの更新を、アトミックにコミットするかアボートするかのいずれかの状態にすることを保証する。すなわち、すべての更新を同時に有効にするのがコミットである。逆に、すべての更新を捨ててトランザクションを実行しなかったのと同じ状態に戻すのがアボートである。また、トランザクション処理システムは、一度トランザクションをコミットしたら、そのトランザクションによる更新が取り消されることがないことも保証する。

【０００３】このようなトランザクション処理システムを実現する方式は広く知られており、例えば“ＴＲＡＮＳＡＣＴＩＯＮ　ＰＲＯＣＥＳＳＩＮＧ：　ＣＯＮＣＥＰＴＳ　ＡＮＤ　ＴＥＣＨＮＩＱＵＥＳ”（Ｊｉｍ　Ｇｒａｙ，Ａｎｄｒｅａｓ　Ｒｅｕｔｅｒ著、Ｍｏｒｇａｎ　Ｋａｕｆｍａｎｎ）にはさまざまな方式が開示されている。

【０００４】トランザクション処理システムにおいて扱いが難しいのは、１つのファイル上のデータを、同時に複数のトランザクションが更新する場合である。１つのファイルには同時に１つのトランザクションしか更新することを許さなければ問題は生じない。しかし、同時に複数のトランザクションを並行に処理して性能を向上させるためには、同時に１つのファイルの中の異なる部分に記録管理するデータを更新できるようにすることが必要になる。この問題は、ファイルではなくデータベースを用いる場合も同様である。以下、ファイルを例に説明するが、データベースにおいても同様である。

【０００５】このような並行に更新処理を行うトランザクション処理システムを実現するために従来から広く用いられているのは、ＷＡＬ（Ｗｒｉｔｅ　Ａｈｅａｄ　Ｌｏｇｇｉｎｇ）と呼ばれる方式である。この方式は、メモリ上のバッファに読み込んだファイルに対し、複数のトランザクションが更新を行った場合、ログファイルにはどのトランザクションがどのファイルのどの部分をどう書き換えたかを記録する。さらに、ログに記録した更新情報が安定記憶装置（例えばハードディスク）上に書かれて消えないことが保証されるまで、バッファ上で行った更新はハードディスクなどの記憶媒体上の元のファイルには書き戻さないようにタイミングを制御する。

【０００６】ＷＡＬ方式ではすべての更新履歴はログに記録されており、システムの障害が発生した場合には、そのログを使ってファイルを正しい状態に戻すことができる。すなわち、障害が発生してシステムを再起動した場合、トランザクション処理システムはログを見て、コミットしたトランザクションが行った更新で、それがファイルに反映されていないものは正しく反映させる。また、アボートしたトランザクションによる更新がファイルに書きこまれていたらそれを取り消す処理を行う。このようなリカバリ処理により、障害発生に対してもトランザクションの原子性（Ａｔｏｍｉｃｉｔｙ）や耐久性（Ｄｕｒａｂｉｌｉｔｙ）や分離性（Ｉｓｏｌａｔｉｏｎ）や一貫性（Ｃｏｎｓｉｓｔｅｎｃｙ）を保証することができる。

【０００７】トランザクション処理システムを実現するもう１つの方式は、シャドウページ方式である。この方式でファイルを更新する場合には、ハードディスク上に、更新前のファイルが書き込まれているページと、更新後のファイルが書き込まれているページの両方を用意し、トランザクションのコミット時にファイルの管理データをアトミックに入れ替える。そのため、常にハードディスク上には正しいファイルが存在することが保証できる。

【０００８】

【発明が解決しようとする課題】従来のＷＡＬ方式によるトランザクション処理方式では、トランザクションがファイルを更新する場合、まず更新内容をログに書いてからファイル本体を更新するため、複数のトランザクションによるファイルの共有を管理するためには複雑な処理手順を必要とする。また、１回の更新に対して、ログとファイル本体への２回の書き込み処理が必要になる。また、リカバリに必要な更新情報をログに記録し、リカバリ時にそのログを使ってファイルを一貫性のある状態に戻す処理は、複雑な処理手順を必要とする。さらに、リカバリ処理は障害発生後の再起動時に集中して行わなければならず、再起動に時間がかかるという問題点がある。

【０００９】一方、シャドウページ方式では、ハードディスク上には常に正しい状態のファイルが存在するの

で、障害発生時のリカバリ処理のオーバヘッドが小さいという特徴を持つ。しかし、シャドウページ方式は、１つのトランザクションのコミット前とコミット後の２つのファイルのデータが記録されたページを、アトミックに交換することを基本操作としている。そのため、複数のトランザクションが同じファイルを更新したい場合には、それらのトランザクションをシリアライズして順に更新して行くようにしなければならない。ＷＡＬ方式のように同じファイルの異なる部分を複数のトランザクションが並行して更新できるようにするためにはシャドウページ方式だけでは無理で、その上にＷＡＬ方式と同じようにログを使ったデータの管理をするなどの工夫が必要になる。そのため、シャドウページ方式は簡単に実装できても、その上に複雑なデータの管理を実装することが必要になるという問題点がある。

【００１０】本発明は、上記事情を考慮してなされたもので、並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することのできるトランザクション処理方法及びトランザクション処理システムを提供することを目的とする。

【００１１】

【課題を解決するための手段】本発明（請求項１）に係るトランザクション処理方法は、バッファ領域上に読み出した同一のファイルを更新対象として複数のトランザクションが処理を行い、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、該一つのコミットするトランザクションによる確定した更新内容と、他のコミットしていないトランザクションによる未確定の更新内容と、該未確定の更新を取り消すための情報とをファイル内容として含む前記ファイルを、安定記憶装置に書き込むことを特徴とする。

【００１２】安定記憶装置とは、例えば、ハードディスク等の、電源を切っても記憶した内容が消えない記憶装置を指す。

【００１３】本発明では、上記のように、あるトランザクションのコミット時に、そのトランザクションが更新したファイルを別のトランザクションも更新している場合には、その別のトランザクションによる更新内容とそれを取り消すための情報も一緒にファイルの中身に書き込んで、そのファイルを安定記憶装置（例えばハードディスク）に書き出すことにより、ファイルとは別にログを用意して複雑な管理を行うようなことをしなくとも、トランザクションの原子性（トランザクション単位で、行った更新の全てが安定記憶装置上で有効になる（コミット）か全て無効になる（アボート）かいずれかの状態にしかならないことを、システムに障害が起きた場合でも保証できること）を担保しつつ、一つのファイルのデータを複数のトランザクションが並行して更新することを実現できる。

【００１４】好ましくは、前記他のコミットしていないトランザクションをコミットする時には、該他のこれからコミットするトランザクションによる新たに確定した更新内容と、以前にコミットしたトランザクションによる既に確定している更新内容のうち該他のこれからコミットするトランザクションによる新たに確定した更新内容とは競合しないものとをファイル内容として含むファイルを、前記安定記憶装置に格納された前記ファイルに上書きするようにしてもよい。

【００１５】あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションをコミットする場合には、後にコミットするトランザクションの更新内容と、先にコミットしたトランザクションの更新内容のうち後にコミットするトランザクションによって書き換えられてはいないものとを書き込んだファイルで、先に安定記憶装置に書き出されたファイル（後にコミットすることになるトランザクションの更新内容とそれを取り消すための情報が入っている）を上書きすれば、トランザクションの原子性を担保できる。なお、あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションをアボートする場合には、アボートするトランザクションによる更新がなかった状態に戻したファイルで、先に安定記憶装置に書き出されたファイル（後にアボートすることになるトランザクションの更新内容とそれを取り消すための情報が入っている）を上書きしても良いし、アボート時には何もしなくとも、下記のように次にファイルを読み出す時にリカバリ処理を行えば、トランザクションの原子性は担保できる。

【００１６】好ましくは、新しくファイルを前記安定記憶装置から読み出した時に、このファイルに、コミットしていないトランザクションによる未確定の更新内容および該未確定の更新を取り消すための情報が含まれているか否かを調べ、含まれている場合には、読み出した前記ファイルを前記情報を用いて前記コミットしていないトランザクションによる更新がなかった状態に戻すようにしてもよい。

【００１７】あるトランザクションがコミットした後、同じファイルを更新している別のトランザクションのコミット／アボートが行われる前に、システムに障害が起きた場合でも、コミットしたトランザクションによる更新は既に安定記憶装置上でファイルに正しく反映されている。これに加えて上記のようなリカバリ処理を行えば、未確定の（別の）トランザクションによる更新はアボートしたものとしてそれを取り消すことができるので、トランザクションの一貫性が保たれる。しかも、このリカバリ処理は、障害回復時に行う必要はなく、次にそのファイルを読み出した時に行えば良い。

【００１８】好ましくは、前記一つのコミットするトランザクションが前記ファイルとは別のファイルをも更新

対象として前記バッファ領域上で処理を行っていた場合には、前記ファイルの前記安定記憶装置への書き込みと同期させて、該一つのコミットするトランザクションによる前記別のファイルに対する確定した更新内容をファイル内容として含む前記別のファイルを前記安定記憶装置に書き込むようにしてもよい。

【００１９】上記のように複数のファイルの書き込みを同期させれば、一つのトランザクションがその処理開始時からコミット時までに複数のファイルに対して行った更新の全てが安定記憶装置上で有効になる（コミット）か、全てが無効になる（アボート）か、いずれかの状態にしかならない（システムに障害が起きた場合でも、一つのトランザクションが行った更新のうち一部分だけが有効になり残りが無効になるということはあり得ない）ことを保証するような、アトミックな書き込みが実現できる。

【００２０】好ましくは、前記ファイルは、一纏まりの情報を分割して記録した複数個のファイルの集合の中の一つのファイルであるようにしてもよい。

【００２１】好ましくは、一纏まりの情報を分割して記録した複数個のファイルの集合に対して二つ以上のトランザクションが前記バッファ領域上で更新を行い、この二つ以上のトランザクションによる更新内容に基づいて、前記バッファ領域上で、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファイルそれぞれの大きさが予め定められた基準に従うように調整し、前記一つのコミットするトランザクションが更新したファイル、および該トランザクションによる更新に伴う前記調整の結果更新あるいは新規に作成されたファイルのうち、少なくとも一つのファイルに対して他のトランザクションが更新を行っている場合には、該少なくとも一つのファイルを更新対象（すなわち前記ファイル）として複数のトランザクションが処理を行ったものとして、前記ファイルの前記安定記憶装置への書き込みを行うようにしてもよい。

【００２２】また、本発明（請求項７）に係るトランザクション処理システムは、トランザクションの処理対象となるファイルを安定記憶装置からバッファ領域上に読み出す手段と、前記ファイルに対して複数のトランザクションのそれぞれが行った更新の内容と該更新を取り消すための情報とを前記バッファ領域上で該ファイル内に書き込む手段と、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイル内に書き込まれた他のコミットしていないトランザクションによる更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容とそのままとした情報とをファイル内容として含む前記ファイルを、前記安定記憶装置に書き出す手段とを備えたことを特徴とする。

【００２３】また、本発明（請求項８）に係るトランザクション処理システムは、トランザクションの処理対象となる一纏まりの情報を分割して記録した複数個のファイルの集合を安定記憶装置からバッファ領域上に読み出す手段と、前記ファイルの集合に対して複数のトランザクションのそれぞれが行った更新の内容に基づいて、前記バッファ領域上で、該更新の内容と該更新を取り消すための情報とをファイル内に書き込むとともに、新しいファイルの追加、ファイル間の情報の移動および不要ファイルの削除のうち少なくとも一つを行うことにより、前記複数個のファイルそれぞれの大きさが予め定められた基準に従うように調整する手段と、前記複数のトランザクションのうちの一つのトランザクションのコミット時に、前記ファイル内に書き込まれた該一つのコミットするトランザクションによる更新の内容については該更新を取り消すための情報を削除することにより確定させ、前記ファイルおよび前記調整の結果更新あるいは新規に作成されたファイルのうちの少なくとも一つのファイルに対して更新を行った他のコミットしていないトランザクションによる、該少なくとも一つのファイル内に書き込まれた更新の内容および該更新を取り消すための情報についてはそのままとし、確定させた内容をファイル内容として含む前記ファイルとそのままとした情報をファイル内容として含む前記少なくとも一つのファイルとを、前記安定記憶装置に書き出す手段とを備えたことを特徴とする。

【００２４】なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

【００２５】また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させるための）プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

【００２６】本発明によれば、並行に動作する複数のトランザクションによるファイルの更新を、簡単な処理手順で実現することができる。本発明によれば、従来のＷＡＬ方式や、シャドウページ方式の上に特殊なログ管理を実装するときのように、データの更新時やコミット処理時あるいはリカバリ時に複雑な処理を必要としない。

【００２７】また、本発明では、障害発生後のリカバリ処理を、再起動時に集中して行う必要は無く、障害で更新途中の状態は次にそのファイルを使う時点で行われるため、リカバリ処理のオーバヘッドが小さい。

【００２８】さらに、本発明では、１つのデータを複数

のファイルに分けて管理することで、ファイルの更新のコミット時にディスクに書き戻すデータの量を減らすことができる。

【００２９】

【発明の実施の形態】以下、図面を参照しながら発明の実施の形態を説明する。

【００３０】最初に従来のファイルシステムについて簡単に説明する。図４３に、従来のトランザクション処理システムの例を示す。従来のトランザクション処理システムでは、トランザクションを処理するアプリケーションプログラム１０１は、データベース管理システム１０２を介してデータの操作を行う。ファイルシステム１０３はハードディスク１０４などの安定記憶装置に記録したファイルの操作や管理を行う。データベース管理システム１０２は、ファイルシステム１０３の管理するファイルを使って、アプリケーションプログラム１０１が必要とするデータを管理する。複数のアプリケーションプログラム１０１が共有しているデータに対する操作の間の並行制御もデータベース管理システム１０２が行う。

【００３１】図１に、本発明の一実施形態に係るトランザクション処理システムの構成例を示す。

【００３２】なお、本実施形態では、安定記憶装置としてハードディスクを用いるものとして説明するが、もちろん安定記憶装置として他の装置を使用することも可能である。また、本実施形態では、ユーザプログラムによりトランザクションが発生するものとして説明するが、もちろんユーザプログラム以外のものがトランザクションを発生させる場合も同様である。

【００３３】本実施形態のトランザクション処理システムでは、図１に示されるように、トランザクションを実行する複数のアプリケーションプログラム２間で共有するファイルの操作は、共有データ管理部４を介して行い、共有しないファイルの操作は、アプリケーションプログラム２がファイルシステム６を直接指示して行う。

【００３４】トランザクションマネージャ８は、各アプリケーションプログラム２が処理するトランザクションの管理を行う。アプリケーションプログラム２は、トランザクションの処理を開始する時点で、トランザクションマネージャ８からトランザクション識別子を割り付けてもらう。以降、トランザクションの処理を進めて行く過程で、共有データ管理部４やファイルシステム６に操作を指示するときには、トランザクション識別子を提示してどのトランザクションによるファイルの更新であるかを知らせる。アプリケーションプログラム２がトランザクションの処理の終了をトランザクションマネージャ８に伝えると、トランザクションマネージャ８は共有データ管理部４やファイルシステム６に対して、トランザクション識別子で指定したトランザクションによるファイルの更新結果をハードディスク１０に書き込んでコミットするか、あるいは更新結果を廃棄してアボートするかを指示する。

【００３５】なお、トランザクションマネージャ８を用いず、ユーザプログラム２が自分でトランザクション識別子に相当する情報を生成したり、コミット（ＣＯＭＭＩＴ）やアボート（ＡＢＯＲＴ）を指示するように実施することもできる（本実施形態では、トランザクションマネージャ８を用いる場合について説明する）。

【００３６】共有データ管理部４は、複数のトランザクションによる同一ファイルの異なる部分の更新を実現する。共有データ管理部４は、（当該共有データ管理部４を介して）どのファイルがどのトランザクションによって更新されているかを、更新管理表４１を用いて管理している。図２に、更新管理表４１の例を示す。更新管理表４１には、ファイル名と、そのファイルを更新中のトランザクションのトランザクション識別子のリストの対応が記録されている。図２では、例えば、Ｆ０１０２というファイルはトランザクション識別子が３６，２１，５の３つのトランザクションによって更新中であることを示している。

【００３７】ファイルシステム６は、トランザクションによるファイルの操作を実現する部分である。ファイルシステム６は、バッファ領域６２を管理するバッファ管理部（図示せず）とトランザクション管理表６１を持っている。ハードディスク１０に記録しているファイルを読んだり更新したりする場合には、そのファイルをバッファ領域６２に読み出してきて、バッファ領域６２上で読んだり更新したりする。新しくファイルを作成する場合には、まずバッファ領域６２にファイルを作成する。バッファ領域６２上で更新や作成されたファイルは、トランザクションのコミット時にハードディスク１０に書き込む。トランザクション管理表６１は、どのトランザクションがどのファイルを更新しているかを管理している。

【００３８】図３に、トランザクション管理表６１の例を示す。トランザクション管理表６１には、トランザクション識別子と、そのトランザクションが更新中のファイルのリストの対応が記録されている。図３の例では、トランザクション識別子“３”のトランザクションはファイルＦ９２６８のみを更新しており、トランザクション識別子“５”のトランザクションはファイルＦ０１００とＦ９２６４の２つのファイルを更新していることを示している。トランザクション管理表６１の最後のエントリは、どのトランザクションにも属していないファイルの更新を管理している。

【００３９】図３に例示したトランザクション管理表６１では、最後の行のトランザクション“ＮＯＴＸ”のエントリがこれに当たる。この例では、ファイルＦ０１０２，Ｆ２１５６，Ｆ３６２４，Ｆ５４９７，Ｆ７５３１は特定のトランザクションの一部として更新されているのではなく、どのトランザクションとも関連せずに更新

されていることを示す。以下で説明するように、共有データ管理部４を介して更新中のファイルは、どのトランザクションとも関連しない状態で更新するので、このエントリに記録される。なお、このエントリに記録されるファイルがどのトランザクションによって更新されているかは、共有データ管理部４の更新管理表４１を参照すれば知ることができる。

【００４０】以下は、データの一例としてXMLドキュメントの形式での情報を例にとり、共有データ管理部４を介して、このXMLドキュメントの形式で情報が記録されているファイルを更新するトランザクション処理システムに本発明を適応した場合についてより具体的に説明する。

【００４１】なお、XMLに関しては、“Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ　（ＸＭＬ）　１．０”（Ｗ３Ｃ　Ｒｅｃｏｍｍｅｎｄａｔｉｏｎ　１０－Ｆｅｂ－９８）に開示されている。

【００４２】この場合、トランザクション処理システムでは、「キー」とそれに対する「値」のペアの集合を、XMLのドキュメントの形式でファイルに記録して管理している。このドキュメントは、キーを指定してその値を検索するという、一般的なデータベースとして使用することができる。

【００４３】図４に、そのようなドキュメントの一例を示す。この例のXMLドキュメントでは、＜ｋｅｙ＞と＜／ｋｅｙ＞のタグで囲まれた部分に「キー」を保持し、＜ｖａｌｕｅ＞と＜／ｖａｌｕｅ＞のタグで囲まれた部分に「値」を保持し、そのペアを＜ｅｎｔｒｙ＞と＜／ｅｎｔｒｙ＞のタグで囲んだエントリを作り、それらが複数集まったものを、最も外の＜ＫＶｔａｂｌｅ＞と＜／ＫＶｔａｂｌｅ＞のタグが囲んでいる。実際には、XMLドキュメントは先頭部分に＜？ｘｍｌ＞で始まるプロローグ部を持つが、ここでは省略する。なお、XMLの用語として、＜ｋｅｙ＞のような開始タグと、＜／ｋｅｙ＞のようなそれに対応する終了タグで囲まれた部分を、エレメントと呼ぶ。

【００４４】図４に例示したXMLドキュメントでは、キー「ＡＰＰＬＥ」に対応する値は「４００円」であり、キー「ＢＩＳＣＵＩＴ」に対応する値は「２５０円」であり、キー「ＣＨＯＣＯＬＡＴＥ」に対応する値は「２００円」であり、キー「ＧＲＡＰＥ」に対応する値は「６００円」になっている。

【００４５】このXMLドキュメントのファイルに対して、トランザクションを実行するアプリケーションプログラム２およびトランザクションマネージャ８は、共有データ管理部４を介して、ＲＥＡＤ（ｔｘｉｄ，ｋｅ＊ｙ）とＵＰＤＡＴＥ（ｔｘｉｄ，ｋｅｙ，ｖａｌｕｅ）とＲＥＭＯＶＥ（ｔｘｉｄ，ｋｅｙ）とＣＯＭＭＩＴ（ｔｘｉｄ）とＡＢＯＲＴ（ｔｘｉｄ）の５種類の操作ができるものとする。本実施形態では、ＲＥＡＤ、ＵＰＤＡＴＥ、ＲＥＭＯＶＥはアプリケーションプログラム２が指示し、ＣＯＭＭＩＴとＡＢＯＲＴはトランザクションマネージャ８が指示するものとする。なお、前述したように、トランザクションマネージャ８が存在せず、アプリケーションプログラム２がＣＯＭＭＩＴやＡＢＯＲＴも指示するように実施することもできる。

【００４６】ＲＥＡＤ（ｔｘｉｄ，ｋｅｙ）は、ｔｘｉｄで指定したトランザクション識別子を持つトランザクションが、ｋｅｙで指定したエントリの値を読み出す操作である。

【００４７】ＵＰＤＡＴＥ（ｔｘｉｄ，ｋｅｙ，ｖａｌｕｅ）は、ｔｘｉｄで指定したトランザクション識別子を持つトランザクションが、ｋｅｙで指定したエントリの値をｖａｌｕｅで指定した値に更新する操作である。ｋｅｙで指定したエントリが無い場合には、新しくエントリを作成する。

【００４８】ＲＥＭＯＶＥ（ｔｘｉｄ，ｋｅｙ）は、ｔｘｉｄで指定したトランザクション識別子を持つトランザクションが、ｋｅｙで指定したエントリを削除する操作である。

【００４９】ＣＯＭＭＩＴ（ｔｘｉｄ）は、ｔｘｉｄで指定したトランザクション識別子を持つトランザクションが行ったファイルの内容の変更を、ディスク上のファイルに書き戻して、変更を永久的なものにする。

【００５０】ＡＢＯＲＴ（ｔｘｉｄ）は、ｔｘｉｄで指定したトランザクション識別子を持つトランザクションが行ったファイルの内容の変更を、すべて取り消して無かったものにする。

【００５１】これらの５つの操作のうち、ＲＥＡＤはファイルの内容を変更しない操作であるのに対し、ＵＰＤＡＴＥとＲＥＭＯＶＥはファイルの内容を変更する操作である。ＣＯＭＭＩＴとＡＢＯＲＴは、トランザクションの処理結果をコミットすなわち正しく完了した状態にするか、アボートすなわち処理結果をキャンセルして処理開始前の状態にする操作である。

【００５２】図４のXMLドキュメントに対して、３つのトランザクション（ｔｘｉｄ＝１のトランザクション（１）、ｔｘｉｄ＝２のトランザクション（２）、ｔｘｉｄ＝３のトランザクション（３）とする）が、以下に示すように並行にアクセスする場合の動作を例に、本実施形態のトランザクション処理システムの動作を説明する。

［１］　ＵＰＤＡＴＥ（１，ＩＣＥＣＲＥＡＭ，４５０円）
［２］　ＵＰＤＡＴＥ（３，ＡＰＰＬＥ，５００円）
［３］　ＲＥＭＯＶＥ（２，ＢＩＳＣＵＩＴ）
［４］　ＵＰＤＡＴＥ（２，ＣＨＯＣＯＬＡＴＥ，３００円）

13　　14

［５］　ＲＥＭＯＶＥ（１，ＧＲＡＰＥ）
［６］　ＵＰＤＡＴＥ（２，ＫＩＷＩＦＲＵＩＴ，３００円）
［７］　ＣＯＭＭＩＴ（３）
［８］　ＣＯＭＭＩＴ（１）
［９］　ＡＢＯＲＴ（２）

本トランザクション処理システムが図４のファイルを更新する場合、まず、共有データ管理部４から指示されたファイルシステム６が、ハードディスク１０にあるファイルの内容をバッファ領域６２に読み出す。トランザクションがコミットするまでは、ファイルの更新はバッファ領域６２にあるコピーに対してのみ行い、ハードディスク１０には書き戻さない。

【００５３】［１］最初に、ＵＰＤＡＴＥ（１，ＩＣＥＣＲＥＡＭ，４５０円）で、トランザクション（１）がキー「ＩＣＥＣＲＥＡＭ」を追加してその値を「４５０円」にする。このとき、ファイルがまだバッファ領域６２に無ければ読み込む。その後、バッファ上のデータに対して、図５に示すようにバッファ領域６２のデータを変更する。ここでは、新しい「ＩＣＥＣＲＥＡＭ」をキーにするエントリを追加し、その値を「４５０円」にする。さらに、このエントリがトランザクション（１）によって更新中であることを示すために、＜ｅｎｔｒｙ＞タグの属性としてＴＸＩＤとＴＸＳＴＡＴＥの２つを記録しておく。属性ＴＸＩＤの値には変更を行ったトランザクションのトランザクション識別子を、属性ＴＸＳＴＡＴＥには変更の種類を記録しておき、ＣＯＭＭＩＴあるいはＡＢＯＲＴの処理時にこの情報を使用する。ここでは、ＴＸＩＤ属性の値としてトランザクション識別子「１」と、ＴＸＳＴＡＴＥ属性の値として更新中を示す「ＵＰＤＡＴＥ」を持たせる。

【００５４】［２］続いて、ＵＰＤＡＴＥ（３，ＡＰＰＬＥ，５００円）で、トランザクション（３）がキー「ＡＰＰＬＥ」の値を「５００円」に変更する。このときは、バッファ領域６２のデータを図６に示すように変更する。ここでは、「ＡＰＰＬＥ」をキーにするエントリの＜ｖａｌｕｅ＞タグを＜ｏｌｄｖａｌｕｅ＞タグに変更して残しておき、新しい値「５００円」を＜ｖａｌｕｅ＞タグで囲ったエレメントを作成してエントリに追加する。さらに、エントリのには、ＴＸＩＤ属性の値としてトランザクション識別子「３」と、ＴＸＳＴＡＴＥ属性の値として更新中を示す「ＵＰＤＡＴＥ」を持たせる。

【００５５】［３］次に、ＲＥＭＯＶＥ（２，ＢＩＳＣＵＩＴ）で、トランザクション（２）がキー「ＢＩＳＣＵＩＴ」のエントリを削除する。このときは、バッファ領域６２のデータを図７に示すように変更する。ここでは、「ＢＩＳＣＵＩＴ」をキーにするエントリの、ＴＸＩＤ属性の値にトランザクション識別子「２」を、ＴＸＳＴＡＴＥ属性の値に削除されたことを示す「ＲＥＭＯＶＥ」を持たせる。

【００５６】［４］次に、ＵＰＤＡＴＥ（２，ＣＨＯＣＯＬＡＴＥ，３００円）でトランザクション（２）がキー「ＣＨＯＣＯＬＡＴＥ」の値を３００円に変更した後のバッファ領域６２のデータは、これまでと同様の処理を行って図８のようになる。

【００５７】［５］次に、ＲＥＭＯＶＥ（１，ＧＲＡＰＥ）でトランザクション（１）がキー「ＧＲＡＰＥ」のエントリを削除した後のバッファ領域６２のデータは、これまでと同様の処理を行って図９のようになる。

【００５８】［６］次に、ＵＰＤＡＴＥ（２，ＫＩＷＩＦＲＵＩＴ，３００円）でトランザクション（２）がキー「ＫＩＷＩＦＲＵＩＴ」に対応するエントリを追加してその値を「３００円」にした後のバッファ領域６２のデータは、これまでと同様の処理を行って図１０のようになる。

【００５９】［７］続いて、ＣＯＭＭＩＴ（３）でトランザクション（３）のコミット処理が起動される。

【００６０】このように複数のトランザクションが同じファイルを更新している場合のコミット処理の方式について説明する。

【００６１】図１０のようにファイルのデータがバッファ領域６２上で変更されている状態で、ＣＯＭＭＩＴ（３）でトランザクション（３）のコミット処理が起動されたとする。このとき、図１１に示すように、トランザクション（３）による変更を、エレメントにＴＸＩＤやＴＸＳＴＡＴＥがついたり＜ｏｌｄｖａｌｕｅ＞タグを持っている一時的な状態から、それらを削除してハードディスク１０に書き戻しても良い正しく更新された状態に変更する。こうしてトランザクション（３）の変更を反映したファイルのデータができると、それを書き戻してハードディスク１０にある元のファイルを置き換える。このとき、本実施形態では、トランザクション（１）やトランザクション（２）による変更途中の状態のデータは、その変更途中のままハードディスク１０に書き戻すことができる（後述するように、変更途中の状態のデータにはその更新を取り消すのに必要な情報を付加しているので、次に読み出したときに更新を取り消すための処理を行うことができる）。

【００６２】［８］次に、ＣＯＭＭＩＴ（１）でトランザクション（１）のコミット処理が起動されたとする。このとき、図１２に示すように、トランザクション（１）による変更を反映させ、それをハードディスク１０に書き戻す。

【００６３】［９］次に、ＡＢＯＲＴ（２）でトランザクション（２）がアボートしたとする。このとき図１３

に示すように、バッファ領域６２にあるファイルのデータからトランザクション（２）による変更を取り消す。

【００６４】以下では、これまでの例で説明した各々の操作の処理手順を説明する。

【００６５】まず、ＲＥＡＤの処理手順について説明する。

【００６６】ＲＥＡＤの操作は指定されたキーに対応するエントリの値を読み出すだけなので簡単である。ただし、ＲＥＡＤの実装には、そのシステムの使い方に応じて２通りの場合がある。他のトランザクションが更新中でなければ値を読み出せるという点に関しては共通であるが、あるトランザクションが読み出したエントリを、そのトランザクションが終了する前に他のトランザクションが更新できるか否かで２通りの方式に分かれる。つまり、１つの方式は、更新できるという解釈で、ＲＥＡＤに関しては何の排他制御もしないという方式である。もう１つの方式は、更新できないと言う解釈で、広く知られたＲＥＡＤ／ＷＲＩＴＥロックのセマンティクスに従って排他制御する方式である。すなわち、同じトランザクションによるＲＥＡＤと更新は両立するが、異なるトランザクションによるＲＥＡＤと更新は両立せず、また、異なるトランザクションのＲＥＡＤどうしは両立するというものである。排他制御をする後者の方式は、排他制御をしない前者の方式を包含するので、ここでは後者の方式の場合の処理手順を説明する。排他制御しない前者の方式は、後者の方式においてＲＥＡＤについては何も制御しないのと等価である。

【００６７】図１４にＲＥＡＤ（ｔｘｉｄ，ｋｅｙ）の処理手順の一例を示す。

【００６８】まず、キーの値がｋｅｙに指定されたものと同じエントリを探す（ステップＳ１１）。

【００６９】エントリが見つかったならば（ステップＳ１２）、そのエントリにＴＸＳＴＡＴＥ属性が付いているか否か、またＴＸＳＴＡＴＥ属性が付いているならばそのＴＸＳＴＡＴＥ属性が「ＲＥＡＤ」「ＵＰＤＡＴＥ」「ＲＥＭＯＶＥ」のいずれであるかを調べる（ステップＳ１３）。

【００７０】ＴＸＳＴＡＴＥ属性が付いていない場合は、誰もアクセスしていないので、そのエントリにＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を付け、ＴＸＩＤ属性の値は「ｔｘｉｄ」に、ＴＸＳＴＡＴＥ属性の値は「ＲＥＡＤ」にし（ステップＳ１４）、＜ｖａｌｕｅ＞部の値を結果として返す（ステップＳ１５）。

【００７１】ＴＸＳＴＡＴＥ属性の値が「ＲＥＡＤ」の場合は、そのＴＸＩＤ属性にまだ「ｔｘｉｄ」が入っていなければ追加し（ステップＳ１６，Ｓ１７）、＜ｖａｌｕｅ＞部の値を結果として返す（ステップＳ１８）。

【００７２】ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、同じトランザクションが変更した値であれば読めるので（ステップＳ１９）、＜ｖａｌｕｅ＞部の値を結果として返す（ステップＳ２０）。そうでない場合には、他のトランザクションが使用中で読めないことを知らせる。

【００７３】ＴＸＳＴＡＴＥ属性の値が「ＲＥＭＯＶＥ」ならば（ステップＳ１３）、無いものを読もうとしているのでエラーになる。

【００７４】なお、エントリが見つからなかったならば（ステップＳ１２）、エラーとなる。

【００７５】図１５に、図４の状態のファイルのデータに対して、ＲＥＡＤ（４，ＣＨＯＣＯＬＡＴＥ）でトランザクション４がキー「ＣＨＯＣＯＬＡＴＥ」の値を読み出した直後のバッファ領域６２上のデータの状態を示す。

【００７６】通常、ＴＸＩＤの値はトランザクション識別子を１つ持っているが、ＴＸＳＴＡＴＥ属性の値が「ＲＥＡＤ」の場合のＴＸＩＤ属性にはトランザクション識別子のリストを持つことが出来る。これは、複数のトランザクションが同時に読み出すことが出来るためである。例えば、トランザクション識別子が１と４と５のトランザクションがＲＥＡＤしている場合、そのＴＸＩＤ属性の値は「１　４５」のようなリストになっている。

【００７７】次に、ＵＰＤＡＴＥの処理手順について説明する。

【００７８】図１６に、ＵＰＤＡＴＥ（ｔｘｉｄ，ｋｅｙ，ｖａｌｕｅ）の処理手順の一例を示す。

【００７９】まず、キーの値がｋｅｙに指定されたものと同じエントリを探す（ステップＳ２１）。

【００８０】エントリが見つかったならば（ステップＳ２２）、そのエントリにＴＸＳＴＡＴＥ属性が付いているか否か、またＴＸＳＴＡＴＥ属性が付いているならばそのＴＸＳＴＡＴＥ属性が「ＲＥＡＤ」「ＵＰＤＡＴＥ」「ＲＥＭＯＶＥ」のいずれであるかを調べる（ステップＳ２３）。

【００８１】ＴＸＳＴＡＴＥ属性が付いていない場合は、誰もアクセスしていないので、そのエントリにＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を付け、ＴＸＩＤ属性の値は「ｔｘｉｄ」に、ＴＸＳＴＡＴＥ属性の値は「ＵＰＤＡＴＥ」にし、現在の＜ｖａｌｕｅ＞部のタグ名を＜ｏｌｄｖａｌｕｅ＞に変更して退避し、新しく＜ｖａｌｕｅ＞部を作ってその値をＵＰＤＡＴＥの第３引数のｖａｌｕｅに指定されたものにする（ステップＳ２４）。

【００８２】ＴＸＳＴＡＴＥ属性の値が「ＲＥＡＤ」の場合は、同じトランザクションだけがＲＥＡＤしている場合に限って更新できる。この場合は（ステップＳ２５）、ＴＸＳＴＡＴＥ属性の値を「ＵＰＤＡＴＥ」に変更し、現在の＜ｖａｌｕｅ＞部のタグ名を＜ｏｌｄｖａｌｕｅ＞に変更して退避し、新しく＜ｖａｌｕｅ＞部を作ってその値をＵＰＤＡＴＥの第３引数のｖａｌｕｅに指定されたものにする（ステップＳ２６）。他のトラン

ザクションがＲＥＡＤ中であれば、ＵＰＤＡＴＥすることができないので、使用中であることを知らせる。

【００８３】ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、同じトランザクションが更新したエントリであれば（ステップＳ２７）、その＜ｖａｌｕｅ＞部の値をＵＰＤＡＴＥの第３引数のｖａｌｕｅに指定されたものに変更する（ステップＳ２８）。他のトランザクションがＵＰＤＡＴＥ中であればＵＰＤＡＴＥすることができないので、使用中であることを知らせる。

【００８４】ＴＸＳＴＡＴＥ属性の値が「ＲＥＭＯＶＥ」の場合は（ステップＳ２３）、無いものを更新しようとしているのでエラーになる。

【００８５】なお、キーの値がｋｅｙに指定されたものと同じエントリが見つからない場合は（ステップＳ２２）、新しくエントリを作成し、＜ｋｅｙ＞部の値と＜ｖａｌｕｅ＞部の値を、それぞれＵＰＤＡＴＥの第２引数と第３引数に指定されたものにしたエントリを作成し、そのＴＸＩＤ属性の値を「ｔｘｉｄ」に、ＴＸＳＴＡＴＥ属性の値を「ＵＰＤＡＴＥ」にする（ステップＳ２９）。

【００８６】次に、ＲＥＭＯＶＥの処理手順について説明する。

【００８７】図１７に、ＲＥＭＯＶＥ（ｔｘｉｄ，ｋｅｙ）の処理手順の一例を示す。

【００８８】まず、キーの値がｋｅｙに指定されたものと同じエントリを探す（ステップＳ３１）。

【００８９】エントリが見つかったならば（ステップＳ３２）、そのエントリにＴＸＳＴＡＴＥ属性が付いているか否か、またＴＸＳＴＡＴＥ属性が付いているならばそのＴＸＳＴＡＴＥ属性が「ＲＥＡＤ」「ＵＰＤＡＴＥ」「ＲＥＭＯＶＥ」のいずれであるかを調べる（ステップＳ３３）。

【００９０】ＴＸＳＴＡＴＥ属性が付いていない場合は、誰もアクセスしていないので、ＴＸＩＤ属性の値を「ｔｘｉｄ」に、ＴＸＳＴＡＴＥ属性の値を「ＲＥＭＯＶＥ」にする（ステップＳ３４）。

【００９１】ＴＸＳＴＡＴＥ属性の値が「ＲＥＡＤ」の場合には、同じトランザクションだけがＲＥＡＤしている場合に限ってＲＥＭＯＶＥできる。この場合は（ステップＳ３５）、ＴＸＳＴＡＴＥ属性の値を「ＲＥＭＯＶＥ」に変更する（ステップＳ３６）。他のトランザクションがＲＥＡＤ中であればＲＥＭＯＶＥすることができないので、使用中であることを知らせる。

【００９２】ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、同じトランザクションが更新したエントリであれば（ステップＳ３７）、ＲＥＭＯＶＥできる。このとき、＜ｏｌｄｖａｌｕｅ＞部があるかどうかで、そのエントリがこのトランザクションで新しく作られたものか、以前から存在していたものかがわかる。＜ｏｌｄｖａｌｕｅ＞部が無い場合には（ステップＳ３８）、

このトランザクションが作成したエントリなので、即座にエントリを削除してしまって良い（ステップＳ４０）。そうでない場合には（ステップＳ３８）、ＴＸＳＴＡＴＥ属性の値を「ＲＥＭＯＶＥ」に変更する（ステップＳ３９）。他のトランザクションがＵＰＤＡＴＥ中であれば（ステップＳ３７）、ＲＥＭＯＶＥすることができないので、使用中であることを知らせる。

【００９３】ＴＸＳＴＡＴＥ属性の値が「ＲＥＭＯＶＥ」の場合は（ステップＳ３３）、無いものを削除しようとしているのでエラーになる。

【００９４】なお、エントリが見つからなかったならば（ステップＳ３２）、エラーとなる。

【００９５】次に、ＣＯＭＭＩＴの処理手順について説明する。

【００９６】図１８に、ＣＯＭＭＩＴ（ｔｘｉｄ）の処理手順の一例を示す。

【００９７】ＣＯＭＭＩＴの処理時には、ファイル中のすべてのエントリに対して図１８の手順で処理を行う。

【００９８】まず、そのエントリのＴＸＳＴＡＴＥ属性の値を調べる（ステップＳ１０１）。

【００９９】ＴＸＳＴＡＴＥ属性を持たない場合は、どのトランザクションもアクセスしていないので何もしなくて良い。

【０１００】ＴＸＳＴＡＴＥ属性が「ＲＥＡＤ」の場合は、そのＴＸＩＤ属性の中にｔｘｉｄが含まれていて（ステップＳ１０２）、ｔｘｉｄ以外のトランザクション識別子も含まれていれば（ステップＳ１０３）、ｔｘｉｄを抜く（ステップＳ１０４）。ＴＸＩＤ属性がｔｘｉｄのみをもっている場合には（ステップＳ１０３）、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除する（ステップＳ１０５）。

【０１０１】次に、ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、そのＴＸＩＤ属性の値がｔｘｉｄと同じであれば（ステップＳ１０６）、＜ｏｌｄｖａｌｕｅ＞部とＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除してその更新を有効にする（ステップＳ１０７）。

【０１０２】ＴＸＳＴＡＴＥ属性がＲＥＭＯＶＥの場合は（ステップＳ１０１）、そのＴＸＩＤ属性の値がｔｘｉｄと同じであれば（ステップＳ１０８）、そのエントリを削除する（ステップＳ１０９）。

【０１０３】すべてのエントリに関して以上の処理が終了すれば、その時点のバッファ領域６２上のデータをハードディスク１０上に書き戻して、ファイルを更新する。

【０１０４】ところで、ＣＯＭＭＩＴ処理の最後に、共有データ管理部４は、バッファ領域６２上で更新したファイルをハードディスク１０に書き込むように、ファイルシステム６に指示する。このとき、このファイルの書き込みは、コミットするトランザクションが更新した他のファイルと同期してアトミックに書き込む必要があ

る。すなわち、トランザクションを実行するアプリケーションプログラム２は、共有データ管理部４を介さずにファイルシステム６に直接指示して複数のファイルの更新を行っている可能性があり、また、共有データ管理部４を介しても複数のファイルを更新している可能性がある（図２および図３参照）。トランザクションとしてアトミックにコミットするためには、これらの複数のファイルへの更新がすべて完全に行われるなければならない。本実施形態のトランザクション処理システムでは、ファイルシステム６は、トランザクション管理表６１によって、アプリケーションプログラム２から直接指示されて更新したファイルのリストをトランザクション毎に管理しており、その情報を用いて、同じトランザクションが更新したファイルを一括してアトミックにハードディスク１０に書き戻すことができる。

【０１０５】しかし、共有データ管理部４を介して操作しているファイルに関しては、ファイルシステム６内ではどのトランザクションとも関連しないファイルとして管理されている。それらのファイルをどのトランザクションのコミット時に書き戻せば良いかは、共有データ管理部４がＣＯＭＭＩＴの処理時に判断する。

【０１０６】そこで、共有データ管理部４がどのトランザクションとも関連せずにオープンして更新したファイルに対して、特定のトランザクションのコミット時に同期してアトミックにハードディスク１０に書き戻すことを指示する機能を、ファイルシステム６に持たせる。ＣＯＭＭＩＴの処理を行う共有データ管理部４は、指定されたトランザクション識別子のトランザクションによる更新をファイルに反映させた後、そのファイルを指定されたトランザクション識別子のトランザクションと同期してハードディスク１０に書き戻すように、ファイルシステム６に指示する。ファイルシステム６は、このようにして指定されたファイルを、指定されたトランザクション識別子のトランザクションのコミット時に、トランザクション管理表６１にあるそのトランザクションが更新したファイルと同時に、アトミックにハードディスク１０に書き戻す。

【０１０７】ただし、ファイルシステム６のコミット処理が終了後は、トランザクションを実行するアプリケーションプログラム２が直接ファイルシステム６に指示して更新したファイルの情報（トランザクション管理表６１のエントリやバッファ領域６２にあるファイルのイメージ）は不要になる。一方、共有データ管理部４が特定のトランザクションに関連せずに更新しているファイルは、その後も他のトランザクションによって更新が続けられるため、どのトランザクションとも関連しない状態のまま共有データ管理部４によって使い続けられる。そのため、ファイルシステム６はそれらのファイルに関する情報は維持しつづける。

【０１０８】次に、ＡＢＯＲＴの処理手順について説明

する。

【０１０９】図１９に、ＡＢＯＲＴ（ｔｘｉｄ）の処理手順の一例を示す。

【０１１０】ＡＢＯＲＴの処理時には、すべてのエントリに対して図１９の手順で処理を行う。

【０１１１】まず、そのエントリのＴＸＳＴＡＴＥ属性の値を調べる（ステップＳ１１１）。

【０１１２】ＴＸＳＴＡＴＥ属性を持たない場合は、どのトランザクションもアクセスしていないので何もしなくて良い。

【０１１３】ＴＸＳＴＡＴＥ属性が「ＲＥＡＤ」の場合は、そのＴＸＩＤ属性の中にｔｘｉｄが含まれていて（ステップＳ１１２）、ｔｘｉｄ以外のトランザクション識別子も含まれていれば（ステップＳ１１３）、ｔｘｉｄを抜く（ステップＳ１０４）。ＴＸＩＤ属性がｔｘｉｄのみをもっている場合には（ステップＳ１１３）、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除する（ステップＳ１１５）。

【０１１４】ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、そのＴＸＩＤ属性の値がｔｘｉｄと同じであれば（ステップＳ１１６）、そのエントリのＵＰＤＡＴＥを取り消す操作を行う。このとき、そのエントリに＜ｏｌｄｖａｌｕｅ＞部があれば（ステップＳ１１７）、現在の＜ｖａｌｕｅ＞部を削除して＜ｏｌｄｖａｌｕｅ＞部のタグ名を＜ｖａｌｕｅ＞に戻すことで更新前の値に戻すとともに、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除する（ステップＳ１１８）。そのエントリに＜ｏｌｄｖａｌｕｅ＞部が無い場合は（ステップＳ１１７）、このトランザクションで新しく作成したエントリであるので、エントリを削除する（ステップＳ１１９）。

【０１１５】ＴＸＳＴＡＴＥ属性がＲＥＭＯＶＥの場合は（ステップＳ１１１）、そのＴＸＩＤ属性の値がｔｘｉｄと同じであれば（ステップＳ１２０）、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除して、ＲＥＭＯＶＥを取り消す（ステップＳ１２１）。

【０１１６】ところで、ＡＢＯＲＴの場合、アボートするトランザクションによるファイルの更新を、バッファ領域６２上で取り消す。このとき、同じファイルを更新していて既にコミットした他のトランザクションがあった場合には、アボートするトランザクションによる更新中の状態もファイルに書きこまれてしまっている可能性がある。しかし、ＡＢＯＲＴ処理時にバッファ領域６２上で取り消した結果のファイルをハードディスク１０に書き込むことは、必ずしも必要無い。なぜなら、次に同じファイルを他のトランザクションが更新してコミットすれば、アボートしたトランザクションによる更新の取り消しはハードディスク１０上に正しく反映される。また、たとえ次にどのトランザクションも更新せずに、そのまま計算機の障害等で忘れ去られてしまった場合で

も、後で述べるリカバリ処理によって、次にそのファイルをオープンする時に、ハードディスク１０上のファイルに残っているコミットしていないトランザクションによる更新途中の状態は取り消される。

【０１１７】次に、リカバリの処理手順について説明する。

【０１１８】本実施形態のトランザクション処理システムでは、コミットしていないトランザクションによって更新途中のファイルの状態がハードディスク１０に書き戻される。そのため、障害で計算機がダウンした場合には、更新途中の状態が残るので、リカバリ処理を行って更新途中で残された変更を取り消さなくてはいけない。従来のトランザクション処理方式では、このようなリカバリ処理は障害が起こって再起動した直後にまとめて行わなければならない。そのため、障害直後の再起動のオーバヘッドは大きい。しかし、本実施形態のトランザクション処理方式では、リカバリ処理は、次にそのファイルをハードディスク１０から読み出す時点で良い。

【０１１９】図２０に、リカバリの処理手順の一例を示す。

【０１２０】共有データ管理部４がまだバッファ領域６２にないファイルを参照するためにオープン処理をファイルシステム６に指示すると、ファイルシステム６はそのファイルをハードディスク１０から読み出してバッファ領域６２にコピーする。その時点で共有データ管理部４は、すべてのエントリに対して図２０に示す手順でリカバリ処理を実行する。

【０１２１】まず、そのエントリのＴＸＳＴＡＴＥ属性を調べる（ステップＳ２０１）。

【０１２２】ＴＸＳＴＡＴＥ属性を持たない場合は、どのトランザクションもアクセスしていないので何もしなくて良い。

【０１２３】ＴＸＳＴＡＴＥ属性の値が「ＲＥＡＤ」の場合は、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除するだけで良い（ステップＳ２０２）。

【０１２４】ＴＸＳＴＡＴＥ属性の値が「ＵＰＤＡＴＥ」の場合は、いずれかのトランザクションが更新している途中で障害が発生したことを示しているので、それらをアボートする。このときの動作は、そのエントリに＜ｏｌｄｖａｌｕｅ＞部があるかどうかで異なる。＜ｏｌｄｖａｌｕｅ＞部があれば（ステップＳ２０３）、そのトランザクションの開始以前から存在していたエントリであるので、まず現在の＜ｖａｌｕｅ＞部を削除し、＜ｏｌｄｖａｌｕｅ＞部のタグ名を＜ｖａｌｕｅ＞に変更して値を元に戻し、さらにＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除する（ステップＳ２０４）。＜ｏｌｄｖａｌｕｅ＞部が無い場合は（ステップＳ２０３）、このトランザクションが作成したエントリであるので、エントリ全体を削除する（ステップＳ２０５）。

【０１２５】ＴＸＳＴＡＴＥ属性の値が「ＲＥＭＯＶＥ」の場合は（ステップＳ２０１）、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性を削除するだけでＲＥＭＯＶＥをキャンセルすることが出来る（ステップＳ２０６）。

【０１２６】ところで、ファイルをディスクから読み出す度にこのようなリカバリ処理を行うのでは、非常にオーバーヘッドが大きい。そこで、ファイルを書き出すときにそのファイルが更新途中の状態を含んでいるかどうかを簡単に識別できる情報を付けておくことで、このオーバヘッドを無くすことが出来る。その方式は、例えば図２１に示すように、ドキュメントのルートのエレメントの属性として、現在いくつのトランザクションがファイルを更新中かを示す情報を持たせる。図２１の例では、ルートである＜ＫＶｔａｂｌｅ＞のＴＸＮＵＭという属性がそれを示しており、現在、トランザクション（１）とトランザクション（２）の２つのトランザクションが更新中であることを示している。こうしておくと、最初にファイルをディスクから読みだして来た時に、そのルートエレメントがＴＸＮＵＭという属性を持っていたら、そのファイル内のすべてのエントリに対して図２０のリカバリ処理を実行する。もしＴＸＮＵＭという属性が無ければ、リカバリ処理は必要無い。

【０１２７】さて、これまでの説明では、エントリはルートエレントである＜ＫＶｔａｂｌｅ＞の下にフラットに記録していたが、例えば良く知られた二分木のような構造を使って、検索を高速化するように実施することができる。図２２は、そのように実施した場合のファイルのデータの例である。この例では、エントリには＜ｋｅｙ＞と＜ｖａｌｕｅ＞以外に、＜ｌｅｓｓ＞と＜ｇｒｅａｔｅｒ＞というエレメントを持つ。そのエレメントのキーより小さなキーを持つエントリは＜ｌｅｓｓ＞の下に、大きなキーを持つエントリは＜ｇｒｅａｔｅｒ＞の下にくるように、二分木を構成している。この場合も、トランザクションによるＵＰＤＡＴＥやＲＥＭＯＶＥなどの更新はフラットな場合と同様に扱うことが出来る。ただし、エントリの追加や削除によって木の構成を組み換える処理が必要になる。この処理は通常の木構造のノードの追加／削除とまったく同じ処理である。木構造の組み換え時には、ＡＶＬ木のような手法を使ってうまく木がバランスするように組み換えたり、アクセス頻度の多いエントリが木のルートに近い場所に来るように組み換えたりするように実施することもできる。

【０１２８】また、本発明は、良く知られたＢ－ＴＲＥＥのような管理構造を使って、ファイルを分割して管理するように実施することも出来る。図２３／図２４／図２５／図２６にその例を示す。ここでは、一つのファイルには最大６個のエントリを収容し、それ以上にエントリが増えるとファイルを分割するというような、Ｂ－ＴＲＥＥと同じアルゴリズムでエントリの追加／削除の処理を行うものとする。図２３／図２４／図２５／図２６の例では、ＦＩＬＥ００１がルートになるファイル、Ｆ

ＩＬＥ００２，ＦＩＬＥ００３，ＦＩＬＥ００４がその下にあるファイルである。

【０１２９】図２７に、このように一纏まりの情報を複数のファイルに分割して記録・管理するようにしたトランザクション処理システムの構成例を示す。

【０１３０】図２７の構成は、図１の構成と比べて、共有データ管理部４が更新グループ表４２を持つ点が異なる。後に説明するように、複数のファイル間で情報を移動してファイル間の構造を変更した場合に、それらのファイルをコミット時に同時にハードディスク１０に書き戻さなければならない。更新グループ表４２は、そのようなコミット時に同時にハードディスク１０に書き戻したりあるいは削除したりする必要のあるファイルのグループを管理するための表である。

【０１３１】図２８に更新グループ表４２の例を示す。この更新グループ表には、コミット時に同時に書き戻したり削除したりすべきファイルのグループ毎に、そのメンバが記録されている。図２８では、例えば、ＦＩＬＥ００１とＦＩＬＥ００２とＦＩＬＥ００５がグループになっている。

【０１３２】さて、図２３／図２４／図２５／図２６のＦＩＬＥ００１から始めて「ＳＥＳＡＭＥ」をキーとするエントリを探す手順は、ＦＩＬＥ００１のルートから始める。まずキーを「ＫＩＷＩＦＲＵＩＴ」と比べるとそれよりも大きい。次にキーを＜ｇｒｅａｔｅｒ＞側の「ＲＯＳＥＭＡＲＹ」と比べるとそれよりも大きい。次に＜ｇｒｅａｔｅｒ＞側を見るとＦＩＬＥ００４へのリンクになっている。そこでＦＩＬＥ００４に移って、キーを「ＴＨＹＭＥ」と比べるとそれよりも小さい。そこで＜ｌｅｓｓ＞側を見ると、キーが「ＳＥＳＡＭＥ」のエントリが見つかる。こうしてエントリが見つかれば、あとはそれへの操作はこれまでの実施形態と同様に、ＴＸＩＤ属性とＴＸＳＴＡＴＥ属性をつけて更新の途中状態を回復可能な形式で保存して行く。

【０１３３】例えば、図２３／図２４／図２５／図２６の状態のデータに対して、トランザクション（１）がキー「ＯＲＡＮＧＥ」のエントリの値を「２５０円」に更新し、さらにキー「ＳＥＳＡＭＥ」のエントリの値を「１５０円」に更新し、それと並行に動いているトランザクション（２）がキー「ＶＯＤＫＡ」のエントリを削除したとする。この状態で図２３／図２４／図２５／図２６のデータは図２９／図３０のような状態になる。ＵＰＤＡＴＥやＲＥＭＯＶＥなどの更新途中の状態の持ち方は、一つのファイルにすべてのエントリを入れる場合と同じであり、更新したエントリにはＴＸＩＤとＴＸＳＴＡＴＥの２つの属性をつけて管理する。

【０１３４】図２９／図３０の状態で、トランザクション（１）と（２）が順にコミットおよびアボートする場合の動作例を説明する。

【０１３５】まず、トランザクション（１）のコミット

時には、トランザクション（１）によって更新されたエントリはファイルＦＩＬＥ００３とＦＩＬＥ００４にのみ存在している。そこで、ＦＩＬＥ００３とＦＩＬＥ００４に対してコミット処理を行った図３１／図３２の状態のデータを作成し、これをトランザクション（１）による他のファイルの更新と同時にハードディスク１０に書き戻す。このとき、ＦＩＬＥ００４にはトランザクション（２）による更新途中の状態が含まれたまま書き戻すのは、これまでの実施形態と同じである。

【０１３６】次に、トランザクション（２）のアボート時には、トランザクション（２）によって更新されたエントリはＦＩＬＥ００４にのみ存在している。そこでＦＩＬＥ００４に対してアボート処理を行った図３３の状態を作成して、アボート処理は終了する。アボート処理時にはファイルのハードディスク１０への書き戻しは必要無いのは、これまでの実施形態と同じである。

【０１３７】さて、この状態で、新しくトランザクション（３）がキー「ＤＯＤＯ」とキー「ＨＯＯＰＯＥ」のエントリを追加し、それぞれの値を「５０００００円」と「９８００００円」にした後のデータの状態を、図３４／図３５／図３６に示す。ここでは、一つのファイルに入れることの出来るエントリの数は決まっているので、この更新例のように一つのファイルにその容量以上のエントリを追加しようとした場合には、ファイル間で組み換えが発生してエントリが移動する。図３４／図３５／図３６の例では、キー「ＤＯＤＯ」とキー「ＨＯＯＰＯＥ」の二つのエントリを追加しようとしたところ、それらはどちらもＦＩＬＥ００２への追加となる。その結果ＦＩＬＥ００２の分割が行われて、新たにＦＩＬＥ００５が作られてエントリが移動している。

【０１３８】このようにファイル間でエントリ情報の移動が発生したり、新しいファイルが追加されたり、あるいは逆にファイルが削除されたりすると、それらのファイルの更新は同時にハードディスク１０へ反映させなければならない。そのために、本実施形態のトランザクション処理システムの共有データ管理部４は、更新グループ表４２を管理している。図３４／図３５／図３６の例のように、ＦＩＬＥ００２へのエントリの追加によってＦＩＬＥ００２の分割が発生し、ＦＩＬＥ００１の更新とＦＩＬＥ００５の作成が発生した場合、共有データ管理部４は更新グループ表４２にＦＩＬＥ００１とＦＩＬＥ００２とＦＩＬＥ００５をひとつのグループとして登録する。

【０１３９】トランザクション（３）のコミットを行う場合、トランザクション（３）が更新したエントリはＦＩＬＥ００１とＦＩＬＥ００５のみに存在するが、更新グループ表４２を参照することで、ＦＩＬＥ００２も同時に書き戻さなければならないことがわかる。そこで、図３７／図３８に示すようにコミット処理を行ったＦＩＬＥ００１とＦＩＬＥ００５に加えてＦＩＬＥ００２の

３つのファイルをハードディスク１０に書き戻す。

【０１４０】このように複数のファイルに分割してデータを持つように本発明を実施する場合には、様々な基準で一つのファイルに入れられるエントリの数を制限することが出来る。そのような基準としては、図３４／図３５／図３６の例のように一つのファイル内のエントリの数で制限する方式、ファイルの物理的なサイズで制限する方式、追加削除の頻度等の情報によって最大のエントリ数を変更する方式、それらの組み合わせなど、様々な方式が考えられる。

【０１４１】本実施形態のように複数のファイルに分割してデータを記憶する方式は、エントリの数が大きくなったときでも、その一部分しか更新していないなら、コミット時にディスクに書き出すデータ量を減らせるという効果がある。すべてのエントリを一つのファイルに入れる方式では、大きなファイルの一部しか変更していない場合でも、コミット時にファイル全体をディスクに書き戻す必要がある。

【０１４２】さて、これまでは、キーと値とのペアというデータベース的な構造を持つデータに対して本発明を適用する場合を例とって説明してきた。以下では、他の例として、より一般的なＸＭＬドキュメントに対して本発明を適用する場合について説明する。

【０１４３】図３９は、一般的なＸＭＬのドキュメントの例である。このドキュメントに対して、トランザクション（１）が「＜ｔｉｔｌｅ＞名古屋＜／ｔｉｔｌｅ＞」の＜ｃｈａｐｔｅｒ＞の＜ａｕｔｈｏｒ＞を「味噌かつ」に変更し、トランザクション（２）が「＜ｃｈａｐｔｅｒ＞＜ｔｉｔｌｅ＞博多＜／ｔｉｔｌｅ＞＜ａｕｔｈｏｒ＞からし明太子＜／ａｕｔｈｏｒ＞＜／ｃｈａｐｔｅｒ＞」を追加した状態のデータが図４０である。

【０１４４】この後、トランザクション（１）のコミット時にデータを図４１の状態に更新し、これをディスクに書き戻す。

【０１４５】さらに、トランザクション（２）のコミット時にはデータを図４２の状態に更新し、これをディスクに書き戻す。

【０１４６】この一般的なＸＭＬドキュメントを対象にした実施形態でも、障害発生によって更新途中で残されたファイルは、次にディスクから読み出した時にチェックし、ＴＸＳＴＡＴＥがＡＤＤのエレメントは削除してＴＸＳＴＡＴＥがＲＥＭＯＶＥのエレメントは残すようなリカバリ処理を行うことで、コミットしていないトランザクションによる一時的な更新を取り消すことが出来る。

【０１４７】本実施形態によれば、並行に動作する複数のトランザクションによるファイルの更新を、簡単な処理手順で実現することができる。本実施形態によれば、従来のＷＡＬ方式や、シャドウページ方式の上に特殊なログ管理を実装するときのように、データの更新時やコミット処理時あるいはリカバリ時に複雑な処理を必要としない。

【０１４８】また、本実施形態では、障害発生後のリカバリ処理を、再起動時に集中して行う必要は無く、障害で更新途中の状態は次にそのファイルを使う時点で行われるため、リカバリ処理のオーバヘッドが小さい。

【０１４９】さらに、本実施形態では、１つのデータを複数のファイルに分けて管理することで、ファイルの更新のコミット時にディスクに書き戻すデータの量を減らすことができる。

【０１５０】なお、以上の各機能は、ソフトウェアとしても実現可能である。

【０１５１】また、本実施形態は、コンピュータに所定の手段を実行させるための（あるいはコンピュータを所定の手段として機能させるための、あるいはコンピュータに所定の機能を実現させるための）プログラムを記録したコンピュータ読取り可能な記録媒体としても実施することもできる。

【０１５２】本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【０１５３】

【発明の効果】本発明によれば、並行に動作する複数のトランザクションによるファイルの更新処理と障害時のリカバリ処理を簡単に効率良く実現することができる。

【図面の簡単な説明】

【図１】本発明の一実施形態に係るトランザクション処理システムの構成例を示す図

【図２】更新管理表の一例を示す図

【図３】トランザクション管理表の一例を示す図

【図４】ＸＭＬドキュメント（データファイル）の一例を示す図

【図５】ＵＰＤＡＴＥ（１，ＩＣＥＣＲＥＡＭ，４５０円）の処理直後のデータの一例を示す図

【図６】ＵＰＤＡＴＥ（３，ＡＰＰＬＥ，５００円）の処理直後のデータの一例を示す図

【図７】ＲＥＭＯＶＥ（２，ＢＩＳＣＵＩＴ）の処理直後のデータの一例を示す図

【図８】ＵＰＤＡＴＥ（２，ＣＨＯＣＯＬＡＴＥ，３００円）の処理直後のデータの一例を示す図

【図９】ＲＥＭＯＶＥ（１，ＧＲＡＰＥ）の処理直後のデータの一例を示す図

【図１０】ＵＰＤＡＴＥ（２，ＫＩＷＩＦＲＵＩＴ，３００円）の処理直後のデータの一例を示す図

【図１１】ＣＯＭＭＩＴ（３）の処理直後のデータの一例を示す図

【図１２】ＣＯＭＭＩＴ（１）の処理直後のデータの一例を示す図

【図１３】ＡＢＯＲＴ（２）の処理直後のデータの一例を示す図

27

【図１４】ＲＥＡＤ（ｔｘｉｄ，ｋｅｙ）の処理手順の一例を示すフローチャート
【図１５】ＲＥＡＤ（４，ＣＨＯＣＯＬＡＴＥ）の処理直後のデータの一例を示す図
【図１６】ＵＰＤＡＴＥ（ｔｘｉｄ，ｋｅｙ，ｖａｌｕｅ）の処理手順の一例を示すフローチャート
【図１７】ＲＥＭＯＶＥ（ｔｘｉｄ，ｋｅｙ）の処理手順の一例を示すフローチャート
【図１８】ＣＯＭＭＩＴ（ｔｘｉｄ）の処理手順の一例を示すフローチャート
【図１９】ＡＢＯＲＴ（ｔｘｉｄ）の処理手順の一例を示すフローチャート
【図２０】リカバリの処理手順の一例を示すフローチャート
【図２１】トランザクションによる更新中情報を持つデータの一例を示す図
【図２２】二分木構造により検索を高速化するデータの一例を示す図
【図２３】複数ファイルに分割したデータ管理方式の一例を示す図
【図２４】複数ファイルに分割したデータ管理方式の一例を示す図
【図２５】複数ファイルに分割したデータ管理方式の一例を示す図
【図２６】複数ファイルに分割したデータ管理方式の一例を示す図
【図２７】本発明の一実施形態に係るトランザクション処理システムの他の構成例を示す図
【図２８】更新グループ表の一例を示す図
【図２９】トランザクション（１）と（２）による更新の例を示す図
【図３０】トランザクション（１）と（２）による更新の例を示す図
【図３１】トランザクション（１）のコミット処理の例＊

28

＊を示す図
【図３２】トランザクション（１）のコミット処理の例を示す図
【図３３】トランザクション（２）のアボート処理の例を示す図
【図３４】トランザクション（３）による更新の例を示す図
【図３５】トランザクション（３）による更新の例を示す図
【図３６】トランザクション（３）による更新の例を示す図
【図３７】トランザクション（３）のコミット処理の例を示す図
【図３８】トランザクション（３）のコミット処理の例を示す図
【図３９】一般的なＸＭＬドキュメントの例を示す図
【図４０】トランザクション（１）と（２）によるデータの更新例を示す図
【図４１】トランザクション（１）コミット後のデータの一例を示す図
【図４２】トランザクション（２）コミット後のデータの一例を示す図
【図４３】従来のトランザクション処理システムの例を示す図
【符号の説明】
２…アプリケーションプログラム
４…共有データ管理部
４１…更新管理表
４２…更新グループ表
６…ファイルシステム
６１…トランザクション管理表
６２…バッファ領域
８…トランザクションマネージャ
１０…ハードディスク

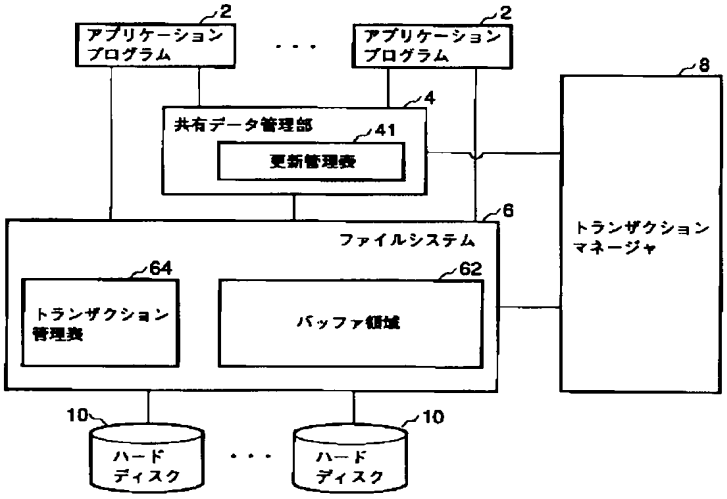
【図２】

ファイル	更新トランザクション
F0102	36,21,5
F2156	6,
F3624	23,5
⋮	
F7531	5,1

【図３】

トランザクション	ファイル
3	F9268
5	F0100,F9264
6	F2164,F1234
21	F0104
⋮	
NOTX	F0102,F2156,F3624,F5497,F7531

【図１】



【図４】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>400円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
</KVtable>
```

【図５】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>400円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図１３】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図６】

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図７】

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図８】

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図９】

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
</KVtable>
```

【図１０】

```
<KVtable>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>APPLE</key>
    <value>500円</value>
    <oldvalue>400円</oldvalue>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
```

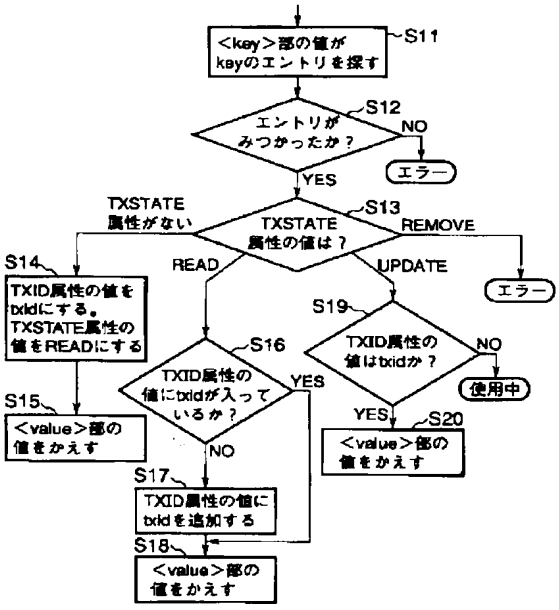
【図１１】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
```

【図１２】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry>
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
```

【図１４】



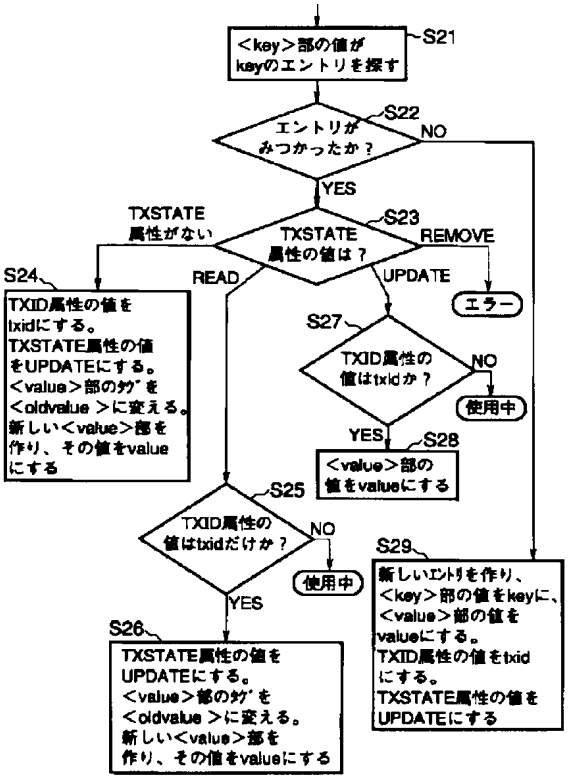
【図２８】

メンバファイル
FILE001,FILE002,FILE005
FILE125,FILE126,FILE131,FILE132
FILE256,FILE328,FILE549
⋮

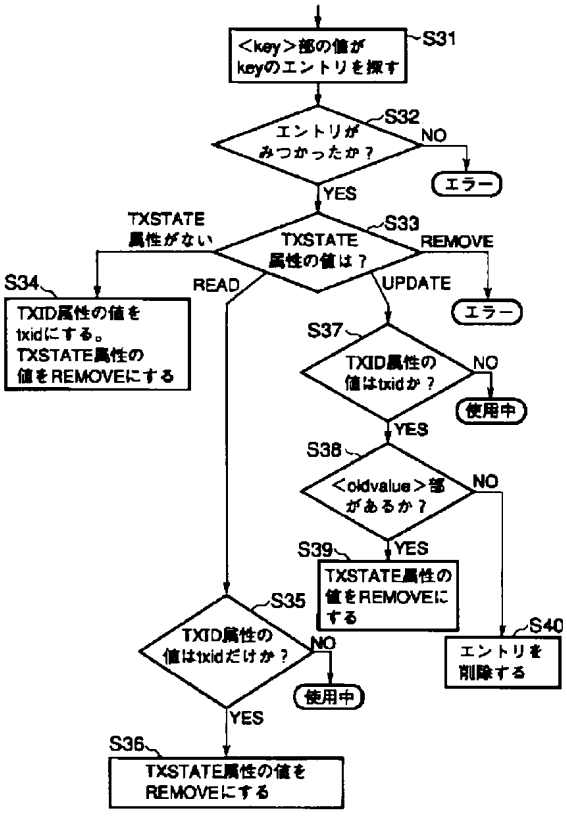
【図１５】

```
<KVtable>
  <entry>
    <key>APPLE</key>
    <value>400円</value>
  </entry>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="4" TXSTATE="READ">
    <key>CHOCOLATE</key>
    <value>200円</value>
  </entry>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
</KVtable>
```

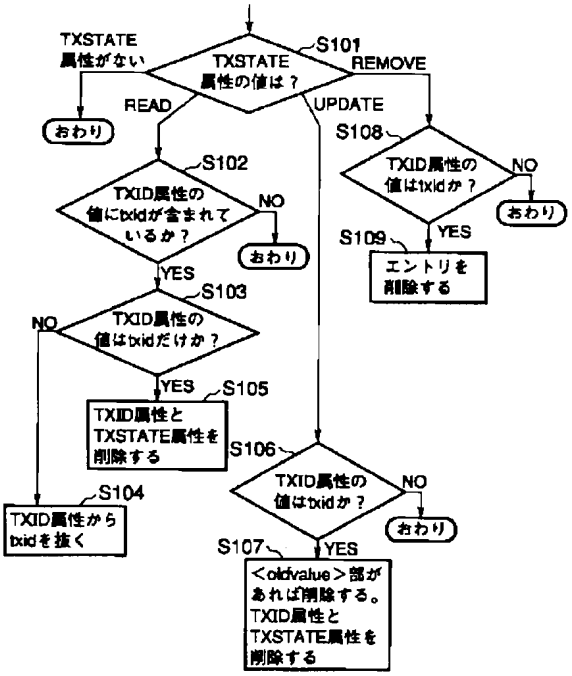
【図１６】



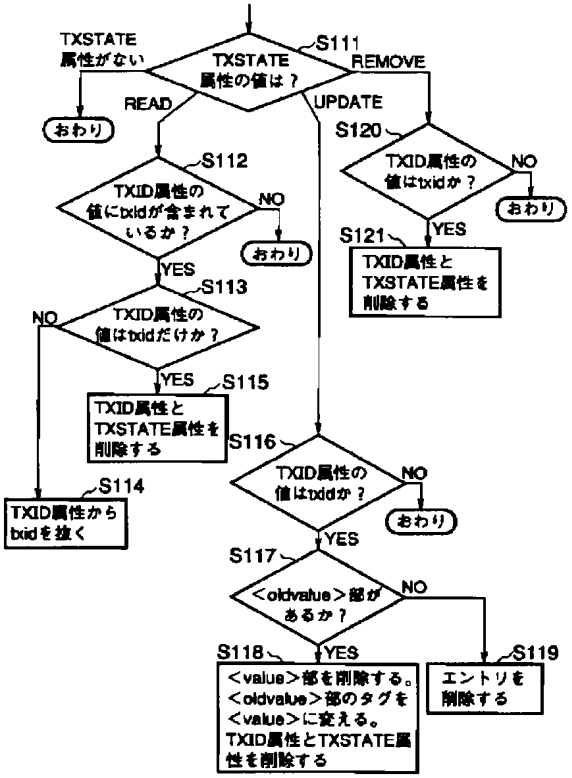
【図１７】



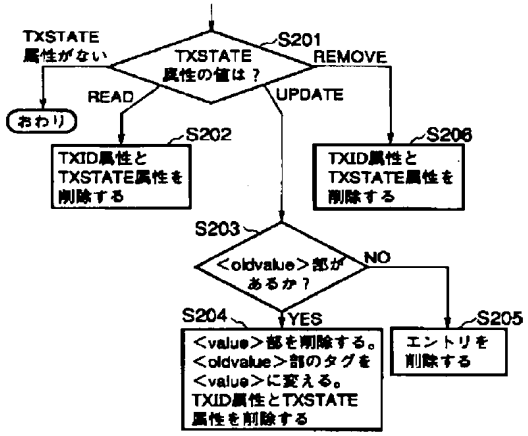
【図１８】



【図１９】



【図２０】



【図２１】

```
<KVtable TXNUM="2">
  <entry>
    <key>APPLE</key>
    <value>500円</value>
  </entry>
  <entry TXID="2" TXSTATE="REMOVE">
    <key>BISCUIT</key>
    <value>250円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>CHOCOLATE</key>
    <value>300円</value>
    <oldvalue>200円</oldvalue>
  </entry>
  <entry TXID="1" TXSTATE="REMOVE">
    <key>GRAPE</key>
    <value>600円</value>
  </entry>
  <entry TXID="1" TXSTATE="UPDATE">
    <key>ICECREAM</key>
    <value>450円</value>
  </entry>
  <entry TXID="2" TXSTATE="UPDATE">
    <key>KIWIFRUIT</key>
    <value>300円</value>
  </entry>
</KVtable>
```

【図２２】

```
<KVlist>
  <entry>
    <key>GRAPE</key>
    <value>600円</value>
    <less>
      <entry>
        <key>BISCUIT</key>
        <value>250円</value>
        <less>
          <entry>
            <key>APPLE</key>
            <value>400円</value>
          </entry>
        </less>
        <greater>
          <entry>
            <key>CHOCOLATE</key>
            <value>200円</value>
          </entry>
        </greater>
      </entry>
    </less>
    <greater>
      <entry>
        <key>KIWIFRUIT</key>
        <value>300円</value>
        <less>
          <entry>
            <key>ICECREAM</key>
            <value>450円</value>
          </entry>
        </less>
        <greater>
          <entry>
            <key>LEMON</key>
            <value>100円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図２３】

```
[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
    <less>
      <link>FILE002</link>
    </less>
    <greater>
      <entry>
        <key>ROSEMARY</key>
        <value>180円</value>
      </entry>
      <less>
        <link>FILE003</link>
      </less>
      <greater>
        <link>FILE004</link>
      </greater>
    </greater>
  </entry>
</KVlist>
```

【図２４】

```
[FILE002]
<KVlist>
  <entry>
    <key>CHOCOLATE</key>
    <value>200円</value>
    <less>
      <entry>
        <key>APPLE</key>
        <value>400円</value>
        <greater>
          <entry>
            <key>BISCUIT</key>
            <value>250円</value>
          </entry>
        </greater>
      </entry>
    </less>
    <greater>
      <entry>
        <key>GRAPE</key>
        <value>600円</value>
        <greater>
          <entry>
            <key>ICECREAM</key>
            <value>450円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

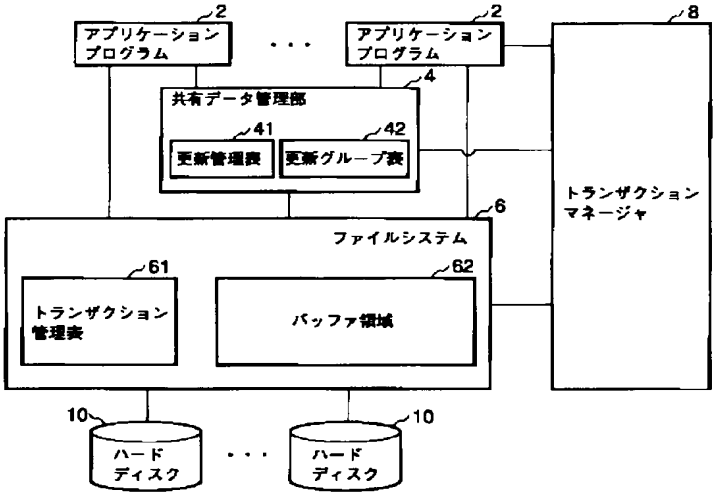
【図２５】

```
[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ORANGE</key>
        <value>210円</value>
        <greater>
          <entry>
            <key>PINEAPPLE</key>
            <value>980円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

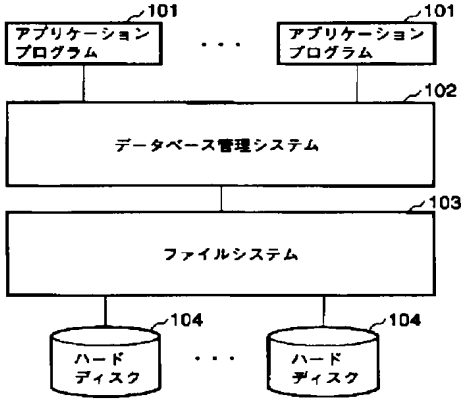
【図２６】

```
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>130円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>VODKA</key>
        <value>1500円</value>
        <greater>
          <entry>
            <key>YOGHURT</key>
            <value>85円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図２７】



【図４３】



【図２９】

```
[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry TXID="1" TXSTATE="UPDATE">
        <key>ORANGE</key>
        <value>250円</value>
        <oldvalue>210円</oldvalue>
        <greater>
          <entry>
            <key>PINEAPPLE</key>
            <value>980円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３０】

```
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry TXID="1" TXSTATE="UPDATE">
        <key>SESAME</key>
        <value>150円</value>
        <oldvalue>130円</oldvalue>
      </entry>
    </less>
    <greater>
      <entry TXID="2" TXSTATE="REMOVE">
        <key>VODKA</key>
        <value>1500円</value>
        <greater>
          <entry>
            <key>YOGHURT</key>
            <value>85円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３１】

```
[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ORANGE</key>
        <value>250円</value>
        <greater>
          <entry>
            <key>PINEAPPLE</key>
            <value>980円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３２】

```
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>150円</value>
      </entry>
    </less>
    <greater>
      <entry TXID="2" TXSTATE="REMOVE">
        <key>VODKA</key>
        <value>1500円</value>
        <greater>
          <entry>
            <key>YOGHURT</key>
            <value>85円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３３】

```
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>150円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>VODKA</key>
        <value>1500円</value>
        <greater>
          <entry>
            <key>YOGHURT</key>
            <value>85円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３４】

```
[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
    <less>
      <entry TXID="3" TXSTATE="UPDATE">
        <key>DODO</key>
        <value>5000000円</value>
      </entry>
      <less>
        <link>FILE002</link>
      </less>
      <greater>
        <link>FILE005</link>
      </greater>
    </less>
    <greater>
      <entry>
        <key>ROSEMARY</key>
        <value>180円</value>
      </entry>
      <less>
        <link>FILE003</link>
      </less>
      <greater>
        <link>FILE004</link>
      </greater>
    </greater>
  </entry>
</KVlist>
```

【図３５】

```
[FILE002]
<KVlist>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
    <less>
      <entry>
        <key>APPLE</key>
        <value>400円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>CHOCOLATE</key>
        <value>200円</value>
      </entry>
    </greater>
  </entry>
</KVlist>

[FILE003]
<KVlist>
  <entry>
    <key>MINT</key>
    <value>120円</value>
    <less>
      <entry>
        <key>LEMON</key>
        <value>100円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ORANGE</key>
        <value>250円</value>
        <greater>
          <entry>
            <key>PINEAPPLE</key>
            <value>980円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３６】

```
[FILE004]
<KVlist>
  <entry>
    <key>THYME</key>
    <value>200円</value>
    <less>
      <entry>
        <key>SESAME</key>
        <value>150円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>VODKA</key>
        <value>1500円</value>
        <greater>
          <entry>
            <key>YOGHURT</key>
            <value>85円</value>
          </entry>
        </greater>
      </entry>
    </greater>
  </entry>
</KVlist>

[FILE005]
<KVlist>
  <entry TXID="3" TXSTATE="UPDATE">
    <key>HOOPOE</key>
    <value>980000円</value>
    <less>
      <entry>
        <key>GRAPE</key>
        <value>800円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ICECREAM</key>
        <value>450円</value>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図３９】

```
<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>海老フライ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
</book>
```

【図４０】

```
<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author TXID="1" TXSTATE="REMOVE">海老フライ</author>
    <author TXID="1" TXSTATE="ADD">味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter TXID="2" TXSTATE="ADD">
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>
```

【図３７】

```
[FILE001]
<KVlist>
  <entry>
    <key>KIWIFRUIT</key>
    <value>300円</value>
    <less>
      <entry>
        <key>DODO</key>
        <value>5000000円</value>
      </entry>
      <less>
        <link>FILE002</link>
      </less>
      <greater>
        <link>FILE005</link>
      </greater>
    </less>
    <greater>
      <entry>
        <key>ROSEMARY</key>
        <value>180円</value>
      </entry>
      <less>
        <link>FILE003</link>
      </less>
      <greater>
        <link>FILE004</link>
      </greater>
    </greater>
  </entry>
</KVlist>
```

【図３８】

```
[FILE002]
<KVlist>
  <entry>
    <key>BISCUIT</key>
    <value>250円</value>
    <less>
      <entry>
        <key>APPLE</key>
        <value>400円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>CHOCOLATE</key>
        <value>200円</value>
      </entry>
    </greater>
  </entry>
</KVlist>

[FILE005]
<KVlist>
  <entry>
    <key>HOOPOE</key>
    <value>980000円</value>
    <less>
      <entry>
        <key>GRAPE</key>
        <value>600円</value>
      </entry>
    </less>
    <greater>
      <entry>
        <key>ICECREAM</key>
        <value>450円</value>
      </entry>
    </greater>
  </entry>
</KVlist>
```

【図４１】

```
<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter TXID="2" TXSTATE="ADD">
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>
```

【図４２】

```
<book>
  <chapter>
    <title>東京</title>
    <author>もんじゃ焼き</author>
  </chapter>
  <chapter>
    <title>名古屋</title>
    <author>味噌かつ</author>
  </chapter>
  <chapter>
    <title>大阪</title>
    <author>たこ焼き</author>
  </chapter>
  <chapter>
    <title>博多</title>
    <author>からし明太子</author>
  </chapter>
</book>
```

フロントページの続き

(72)発明者　岐津　俊樹
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(72)発明者　前田　誠司
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(72)発明者　矢尾　浩
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(72)発明者　矢野　浩邦
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

Ｆターム(参考)　5B082 GB04